Efficient Workload Consolidation for Composable/Disaggregated Data Centers Considering Migration Cost

Chao Guo* and Moshe Zukerman

Department of Electrical Engineering, City University of Hong Kong, Hong Kong SAR,China

**Corresponding e-mail: chaoguo6-c@my.cityu.edu.hk*

ABSTRACT

Resource disaggregation significantly improves resource flexibility, efficiency, and availability in Data Centers (DCs) by decoupling different resource modules in servers and interconnecting them with high-speed networking technologies, e.g., optical networking, leading to Composable/Disaggregated DCs (CDCs or DDCs). CDC also brings benefits to workload migration, a technique widely used to consolidate workloads to fewer nodes to save energy. CDC may reduce migration costs as a workload can be partly migrated instead of entirely as in a server-based DC (SDC). We consider the problem of workload consolidation in a CDC with objectives to minimize both the number of active nodes and the number of migrated elements. An integer linear programming formulation that uses a weighted sum approach is provided to address this two-objective problem. Results show that higher resource utilization and lower migration costs can simultaneously be achieved in a CDC as compared to an SDC.

Keywords: Composable/disaggregated infrastructure, data center, ILP, consolidation, migration

1. INTRODUCTION

Composable/Disaggregated Infrastructure (CDI) is emerging as a new hardware architecture for Data Centers (DCs) which utilizes high-speed interconnection technologies, e.g., optical networking [1][2], to aggregate different types of resources into shared resource pools [3]. A DC employing CDIs is often referred to as a Composable DC (CDC) [4] or Disaggregated DC (DDC) [1]. CDCs can achieve high flexibility, efficiency, and availability in resource allocation thanks to the decoupling of different resources and resource pooling [1]. It may also reduce the high cost of workload migration, which is a technique widely used in different scenarios, e.g., "server consolidation, zero-downtime hardware maintenance, energy management, and traffic management" [5]. In a server-based DC, migrating a workload requires moving the entire workload from its original host to another server. This process involves significant data movement, leading to network overheads and service interruption [5]. On the contrary, CDCs enable partial migration, which will help reduce the overheads.

We consider the problem of workload consolidation in CDCs, which involves a large number of workload migrations. Workload consolidation is effective for DCs to solve the resource fragmentation problem caused by the unexpected arrival and departure of services. It re-arranges workloads that spread over many hardware nodes to the fewest ones to save energy. There are many publications in the literature focusing on resource allocation for CDCs considering different criteria, such as resource utilization and energy efficiency [4], [6]-[8], networking performance [9]-[12], and service reliability [13]-[14]. However, they never considered the challenges in workload consolidation, i.e., the significant migration cost incurred in moving workloads from their original hosts to other hosts. In this paper, we aim to minimize both the number of active nodes and the number of migrated elements in the workload consolidation problem. We provide an integer linear programming (ILP) formulation for this problem. We also provide approximate Pareto fronts by varying the weight when solving the ILP problem. Our results show that higher resource utilization and lower migration costs can simultaneously be achieved in a CDC than in an SDC. Moreover, it shows that a larger disaggregation scale can further improve the optimal solution.

2. WORKLOAD CONSOLIDATION FOR A CDC

2.1 CDC Architecture

Fig. 1 illustrates a CDC architecture as compared with an SDC architecture. Fig. 1(a) shows the SDC architecture, where each node is a server integrating different types of resource modules. Under this architecture, when a new type of resource needs to be introduced, e.g., a new GPU type, a new type of server is needed to equip this resource. Fig. 1(b) shows the CDC architecture, where each node contains one primary and secondary resource, e.g., each node needs additional processing and memory resources reserved for the controller or monitor. Besides, each computing node shown in the figure also contains several DRAMs playing the role of local memory. As widely discussed, completely disaggregating memory from processors is too difficult and even infeasible due to the stringent requirements of processor-memory communications. Thus, it is necessary to reserve memory in each computing node. Different from some publications which regard the local memory as a local cache [3] that is not involved in resource allocation, we follow the industry [15], taking local memory as an allocable resource while regarding the memory in memory nodes as a memory extension.

Due to the high bandwidth and low latency communication requirements between different resource modules, it is not practical to aggregate resources in a large-scale DC into a single resource pool [1]. A practical approach is

to divide the modules into multiple pools. A request can flexibly select modules from one pool to compose a system but cannot select modules from different pools. The size of each pool, or disaggregation scale, is a parameter related to the number of nodes contained in each pool. For example, a typical disaggregation scale is the rack scale, where resource modules in one or several racks form a pool [1]. Recent studies on Microsoft Azure DC applications have suggested a disaggregation scale of 8 to 32 CPU sockets (equivalent to 8 to 32 blade servers) to ensure acceptable performance levels [15]. In this study, we will also consider the impact of the disaggregation scale on the performance of workload consolidation in CDCs.

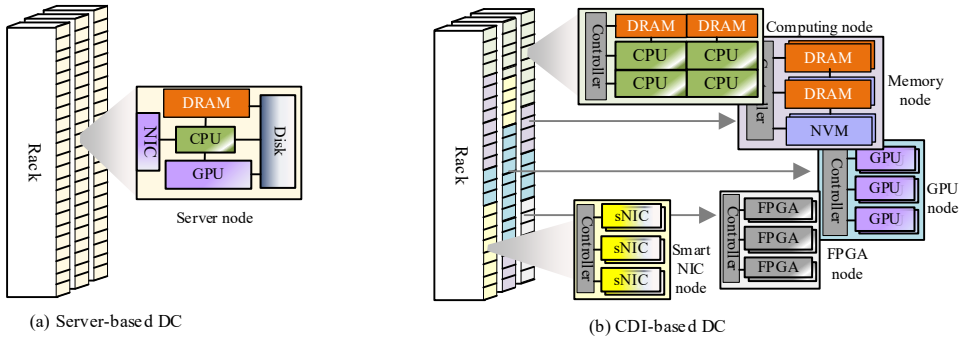

Fig. 1 Illustration of server-based vs. CDI-based DCs.

2.2 Problem Description

Consider a CDC consisting of different types of nodes arranged in multiple resource pools. Each node contains one or multiple types of resources with certain capacities. We are also given a set of workloads and the current placement of each workload. Each workload uses multiple nodes, and we refer to the resources provisioned by one node as an *element*. An element is conceptually similar to a VM, as a VM is an abstraction of the resources assigned to it. We also assume that each workload can use at most one node for each node type. We aim to consolidate all the workloads spreading over the CDC to the fewest nodes, with two objectives: 1) Minimize the number of *active* nodes; 2) Minimize the number of migrated elements. Additionally, we require that the amount of resources provisioned by a node should be equal to the amount obtained from the same type of node before the consolidation.

2.3 ILP Formulation

The sets, parameters, and decision variables used in the formulation are as follows.

Sets:

$\boldsymbol{T}$	Set of node types.
$\boldsymbol{R_t}$	Set of resource types provided by each node of node type t, e.g., a compute node provides (CPU) cores and local memory.
$\boldsymbol{N_t}$	Set of nodes of node type t.
$\boldsymbol{V}$	Set of workloads.
$\boldsymbol{P}$	Set of resource pools.

Parameters:

C_{nt}^{r}	The capacity of resource r ($r \in \boldsymbol{R_t}$) in node n ($n \in \boldsymbol{N_t}$) of node type t ($t \in \boldsymbol{T}$).
π_{nt}^{p}	Binary parameter indicating whether node n of node type t is in pool p ($p \in \boldsymbol{P}$).
η_{nt}^{v}	Binary parameter indicating whether workload v ($v \in \boldsymbol{V}$) currently uses node n of node type t.
D_{rt}^{v}	The amount of resource r allocated by a certain node of node type t to workload v.
θ	Weight factor, $\theta \in [0,1]$.

Decision variables:

x_{nt}^{v}	Binary variable that equals one if workload v uses node n of node type t after the consolidation; zero, otherwise.
y_{vp}	Binary variable that equals one if workload v uses pool p after the consolidation; zero, otherwise.
z_{nt}	Binary variable that equals one if node n of node type t is active after the consolidation; zero, otherwise.
N^{Act}	Integer variable denoting the number of active nodes after the consolidation.
N^{Mig}	Integer variable denoting the number of migrated elements.

The objective function and constraints are as follows.

Objective function:

$$\text{minimize: } \theta \cdot N^{Act} + (1-\theta) \cdot N^{Mig} \tag{1}$$

Constraints:

$$\sum_{n \in \boldsymbol{N_t}} x_{nt}^{v} = \sum_{n \in \boldsymbol{N_t}} \eta_{nt}^{v} \ \forall v \in \boldsymbol{V}, \boldsymbol{t} \in \boldsymbol{T} \tag{2}$$

$$z_{nt} \leq \sum_{v \in \boldsymbol{V}} x_{nt}^{v} \ \forall n \in \boldsymbol{N_t}, t \in \boldsymbol{T} \tag{7}$$

$$\sum_{p \in \boldsymbol{P}} y_{vp} = 1 \; \forall v \in \boldsymbol{V} \quad (3)$$

$$y_{vp} \leq \sum_{n \in \boldsymbol{N_t}} x_{nt}^{v} \cdot \pi_{nt}^{p} \; \forall v \in \boldsymbol{V}, t \in \boldsymbol{T}, p \in \boldsymbol{P} \quad (4)$$

$$y_{vp} \geq x_{nt}^{v} \cdot \pi_{nt}^{p} \; \forall v \in \boldsymbol{V}, n \in \boldsymbol{N_t}, t \in \boldsymbol{T}, p \in \boldsymbol{P} \quad (5)$$

$$z_{nt} \geq x_{nt}^{v} \; \forall n \in \boldsymbol{N_t}, t \in \boldsymbol{T}, v \in \boldsymbol{V} \quad (6)$$

$$\sum_{v \in \boldsymbol{V}} D_{rt}^{v} \cdot x_{nt}^{v} \leq C_{nt}^{r} \; \forall r \in \boldsymbol{R}, n \in \boldsymbol{N_t}, r \in \boldsymbol{R_t}, t \in \boldsymbol{T} \quad (8)$$

$$N^{Act} = \sum_{n \in \boldsymbol{N_t}, t \in \boldsymbol{T}} z_{nt} \quad (9)$$

$$N^{Mig} = \sum_{v \in \boldsymbol{V}, n \in \boldsymbol{N_t}, t \in \boldsymbol{T}} \eta_{nt}^{v} \cdot (1 - x_{nt}^{v}) \quad (10)$$

Explanation

We aim to minimize the objective function (1) which is the weighted sum of the number of active nodes and the number of migrated elements after the consolidation. Constraint (2) ensures that if a workload uses a node of a certain node type, i.e., $\sum_{n \in \boldsymbol{N_t}} \eta_{nt}^{v} = 1$, the workload also uses a node of this node type after the consolidation, i.e., $\sum_{n \in \boldsymbol{N_t}} x_{nt}^{v} = 1$, and if the workload does not use a certain type of node, it cannot use this node type after the migration. Constraint (3) ensures that a workload can only use one resource pool. Constraint (4) ensures that a workload does not use a pool if the workload does not use nodes from this pool. Constraint (5) ensures that a workload uses a pool as long as the workload uses nodes from this pool. Constraint (6) ensures that a node is active as long as there are workloads using this node. Constraint (7) ensures that a node is inactive when no workload uses this node. Constraint (8) ensures that the capacity of each node is not violated. Constraint (9) ensures the number of active nodes is correctly calculated. Constraint (10) ensures that the number of migrated elements is correctly calculated by summing up the number of elements whose new placement differs from their initial placement, i.e., $\eta_{nt}^{v} \cdot (1 - x_{nt}^{v}) = 1$.

3. Evaluation

3.1 Parameter Settings

We considered five types of nodes, i.e., $\boldsymbol{T}$ ={Computing, Memory, A1, A2, A3}, where A1, A2, A3 represent three types of accelerators. There are 90 nodes, including 30 computing nodes, 30 memory nodes, 10 A1 nodes, 10 A2 nodes, and 10 A3 nodes. Each computing node contains 48 (CPU) cores and 192 GB of local memory. Each memory node contains 1536 GB of memory. Each of A1, A2, and A3 nodes contains 60 A1, A2, and A3 units, respectively. We considered three cases with different disaggregation scales, resulting in different pools: 1, 5, and 10. In each of the three cases, all nodes of each node type are equally divided. For example, in the 5-pool case, each pool contains 6 computing, 6 memory, 2 A1, 2 A2, and 2 A3 nodes. We also considered the server-based architecture as a baseline case besides the three CDC cases. There are 90 servers. Each server contains 16 cores, 64 GB of local memory, and 512 GB of memory. Note that we still differentiate the memory in each server into two parts, i.e., local memory and memory. This setting is for the fair comparison between the two architectures concerning the number of migrated elements, as they are from different types of nodes in the CDC cases. Among the 90 servers, 30 servers contain 20 A1 units each, another 30 servers with 20 A2 units each, and the rest 30 servers with 20 A3 units each. With these settings, the total capacity of each resource type in the SDC case is equal to that in the CDC cases.

Each workload requires a certain amount of cores, local memory, memory, and one of the three accelerators. The required amount of resources are randomly generated, with U[1, 12] of cores, U[1, 48] GB of local memory, U[32, 384] GB of memory, and U[1, 15] units of A1, A2, or A3. U[a, b] is a random integer value that varies from a to b (with a and b included). Workloads are assumed to have the same probability of requiring A1, A2, or A3. We considered 120 workloads. We use Java to build a simulation environment and apply random allocation methods to accommodate all workloads, with all nodes being active before the consolidation. This allocation result is then used as the input for our problem.

We use the commercial solver AMPL/Gorubi to solve the ILP formulations. We also extend the formulation to the SDC case by a simple modification to constraint (10), i.e., introduce factor 3 to the calculation of N^{mig}, as each server contains three elements for each workload.

3.2 Numerical Results

Fig. 2 gives the approximate Pareto fronts obtained by solving the ILP formulation with varied weights in the objective function, and Fig. 3 gives the number of migrated elements changing with the number of saved active nodes. We first observe that, for all four test cases, reducing the number of active nodes leads to an increasing number of migrated elements, which illustrates the trade-off between the efficiency improvement and migration cost in workload consolidation. We further observe from Fig. 2 that the approximate Pareto fronts of the three CDC cases are significantly lower than that of the SDC, which implies that CDC, regardless of the disaggregation scale, performs better than SDC. We see from Fig. 3 that CDC can save 40 active nodes at most, showing a 29% improvement compared to the SDC, which can save 31 active nodes at most. Furthermore, this improvement is

achieved with fewer migrated elements. The three CDC cases have 136, 124, and 105 migrated elements, all fewer than the SDC, which has 138 migrated elements. These results show that the CDI-based architecture can simultaneously achieve higher resource efficiency and less migration cost than the server-based architecture.

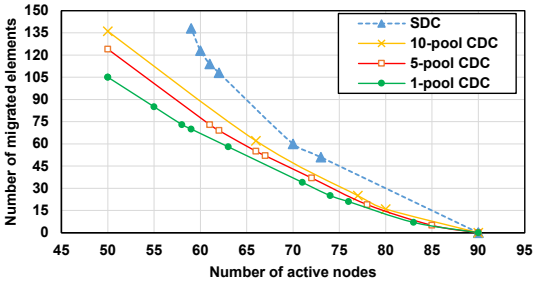

Figure 2. Approximate Pareto fronts (number of migrated elements vs. number of active nodes).

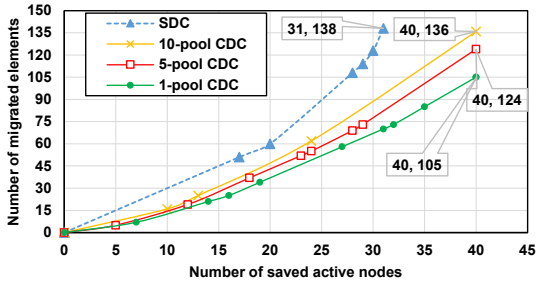

Figure 3. The number of migrated elements vs. the number of saved active nodes.

When comparing the three CDC cases, we find that the 1-pool CDC performs the best, followed by the 5-pool CDC, and finally, the 10-pool CDC. This trend shows that the larger the disaggregation scale, the more benefits can be obtained by the CDI.

4. CONCLUSIONS

We have provided an ILP formulation for workload consolidation in CDCs to minimize the number of active nodes and the number of migrated elements. The results have shown that CDCs with different disaggregation scales can save more active nodes with fewer migrated elements as compared to SDCs. In addition, a CDC with a larger disaggregation scale has fewer migrated elements than a CDC with a smaller scale.

ACKNOWLEDGEMENT

This work was supported by a grant from the City University of Hong Kong, Hong Kong SAR (CityU 9610544).

REFERENCES

[1] R. Lin, Y. Cheng, M. De Andrade, L. Wosinska, and J. Chen: Disaggregated data centers: Challenges and trade-offs, *IEEE Commun. Mag.*, vol. 58, no. 2, pp. 20-26, 2020.

[2] K. Tokas *et al.*: Slotted TDMA and optically switched network for disaggregated datacenters, in *Proc. ICTON,* Girona, Spain, 2017, pp. 1-5.

[3] Bowers, D., M. Reynolds, and C. Dekate: Hype cycle for compute infrastructure, 2018. 2018, Gartner.

[4] O. O. Ajibola, T. E. El-Gorashi, and J. M. Elmirghani: Energy efficient placement of workloads in composable data center networks, *J. Lightwave Technol.*, vol. 39, no. 10, pp. 3037-3063, May 2021.

[5] F. Zhang, G. Liu, X. Fu, and R. Yahyapour: A survey on virtual machine migration: Challenges, techniques, and open issues, *IEEE Commun. Surv. Tutor.*, vol. 20, no. 2, pp. 1206-1243, Secondquarter 2018.

[6] H. M. Mohammad Ali *et al.*: Energy efficient resource provisioning with VM migration heuristic for disaggregated server design, in *Proc. ICTON*, Trento, Italy, 2016, pp. 1-5.

[7] H. M. Mohammad Ali *et al.*: Future energy efficient data centers with disaggregated servers, *J. Light. Technol.*, vol. 35, no. 24, pp. 5361-5380, Dec. 2017.

[8] A. Pagès *et al.*: On the benefits of resource disaggregation for virtual data centre provisioning in optical data centres, *Comput. Commun.*, vol. 107, pp. 60-74, Jul. 2017.

[9] A. Pagès, F. Agraz, and S. Spadaro: Analysis of service blocking reduction strategies in capacity-limited disaggregated datacenters, in *Proc. OFC*, San Diego, CA, USA, 2020, pp. 1-3.

[10] X. Guo *et al.*: RDON: A rack-scale disaggregated data center network based on a distributed fast optical switch, *J. Opt. Commun. Netw.*, vol. 12, no. 8, pp. 251-263, Aug. 2020.

[11] G. Zervas *et al.*: Optically disaggregated data centers with minimal remote memory latency: Technologies, architectures, and resource allocation, *J. Opt. Commun. Netw.*, vol. 10, no. 2, pp. A270-A285, Feb. 2018.

[12] O. O. Ajibola, T. E. H. El-Gorashi, and J. M. H. Elmirghani: A network topology for composable infrastructures, in *Proc. ICTON*, Bari, Italy, 2020, pp. 1-4.

[13] C. Guo *et al.*: Exploring the benefits of resource disaggregation for service reliability in data centers, *IEEE Trans. on Cloud Comput.*, to be published.

[14] C. Guo, M. Zukerman, and T. Wang: Radar: Reliable resource scheduling for composable/disaggregated data centers, *IEEE Trans Ind. Informat.*, to be published.

[15] Li, Huaicheng, *et al.*: First-generation memory disaggregation for cloud platforms, *arXiv preprint arXiv*:2203.00241, 2022.